Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática

INFORME LABORATORIO Nº3
PARADIGMA ORIENTADO A OBJETOS (JAVA)

Fecha: 11/12/2023
Nombre: Thomas Gustafsson Cortés.
Profesor: Roberto González.
Sección: A-1.

1. Introducción

El presente informe está compuesto por una descripción del problema a abordar, una descripción del paradigma utilizado, seguido de un análisis del problema en profundidad, un diseño de solución para el problema anteriormente mencionado, aspectos de la implementación, instrucciones de uso, resultados finales y conclusiones. Esto es para lograr abordar el tema del nuevo paradigma utilizado con la respectiva problemática presentada.

Quiero aclarar primeramente que se explicará la implementación de un sistema simplificado que pueda crear, desplegar y administrar un chatbot. Utilizando el paradigma orientado a objetos (POO) y el lenguaje de programación JAVA.

Siendo POO un tipo de paradigma imperativo, basándose principalmente en objetos como su nombre lo describe, utilizando también clases y sus relaciones entre sí.

Por otro lado, nace el problema de poder lograr una abstracción únicamente “orientada a objetos” de lo solicitado, además de la complicación de poder entender e

implementar este nuevo paradigma eficientemente, pero este último punto se puede amortiguar al saber que es bastante parecido a paradigmas anteriormente estudiados.

2. Descripción del problema

El problema nace desde la necesidad de poder crear un sistema de chatbots ITR (Respuesta de Interacción a Texto) simplificada. El sistema en su fase final debe de poder lograr una fluida comunicación/interacción entre el usuario y el chatbot, y de un chatbot con otro. Siempre girando en todo momento en torno a la utilización del paradigma orientado a objetos en Java, que es ahí donde recae el verdadero problema, ya que si bien el paradigma facilita en gran manera al momento de utilizar clases con sus respectivas características, pero el nivel de abstracción debe de ser mucho mayor, añadiendo que el punto de vista frente a la necesidad anteriormente planteada, es completamente diferente comparado al paradigma lógico y funcional, que fue utilizado para los laboratorios anteriores. Si bien este problema es bien limitado al tener en cuenta que solamente se puede utilizar un solo paradigma, pero ya que es lo justo y lo necesario que se busca aprender, su alcance cubre por completo el lograr entender e implementar bien el paradigma orientado a objetos frente a la necesidad de crear un sistema de chatbots ITR.

3. Descripción del paradigma

El paradigma orientado a objetos (POO) consiste en un tipo de paradigma imperativo donde se basa en gran medida en el concepto de “objetos”, siendo estos “objetos” una abstracción de entidades de la realidad, donde estas pueden ser concretas o abstractas. Estos objetos pueden poseer varias características llamadas “atributos”, también pueden poseer comportamientos, siendo este último llamado “métodos”. Los programas pueden lograr crear objetos, con las características anteriormente mencionadas, guardándolas en memoria, logrando así el poder interactuar entre sí.

Existen objetos que pueden ser de distinta clase, o sea, de distinta tipo, y estos pueden tener diferencias entre sus atributos y métodos nombrados anteriormente. Añadir que los métodos pueden lograr leer o escribir los atributos del mismo objeto, ya que estos tienen una noción de si mismos. De esta forma pueden lograr comunicarse entre si mediante un paso de mensajes. (Gonzalez, 2023)

4. Análisis del problema

Los requisitos específicos que se deben de cubrir son el de crear un ambiente contenedor de chatbots, crear chatbots donde se puedan identificar, añadir preguntas, opciones y poder enlazarlos mediante flujos hacia otros chatbots. Interacción con el chatbot mediante opciones y textos considerando palabras claves y ofrecer mediante el chatbot una síntesis de las interacciones hechas. Siempre utilizando la mirada del paradigma orientado a objetos.

También hay funciones específicos a cubrir, estos son:

- **option:** Crea opciones a escoger, esta construye una opción para un flujo de un chatbot, siendo estos únicos mediante su propio id. Cada opción se une a un chatbot y un flujo específicos.
- **flow:** Función que construye un flujo de un chatbot, siendo este identificado mediante un id, el clase también verifica que las opciones agregadas no se repitan, esto es verificando por medio de sus id.
- **flowAddOption:** Modifica un flujo para poder añadirle una nueva opción, pero primero verifica que no esté repetida mediante su id, si está repetida la opción, no es agregada.
- **chatbot:** Función que crea un chatbot con un id único, se comprueba la duplicidad al añadirlo en un sistema, también esta clase verifica que los flujos añadidos no se repitan comparándolos utilizando el id de cada uno.
- **chatbotAddFlow:** Añade un nuevo flujo a un chatbot, verificando que este no se repita, esto se valida mediante el id del flujo, si es así, no se añade.
- **User:** Función que construye un usuario que puede ser un usuario común o un usuario administrador, cada uno con métodos diferentes (aunque no todos), estos se validad por medio de su nombre, como un nombre único.
- **system:** Función que construye un sistema de chatbots. Este también contiene el historial del chat de cada usuario, aclarar que tiene el String formateado de cada mensaje del usuario y chatbot.
- **systemAddChatbot:** Función que añade un nuevo chatbot a la lista de chatbots de un sistema en específico, pero primero verifica que este nuevo chatbot, por medio de su id, no se repita.
- **systemAddUser:** Añade un nuevo usuario al sistema, verificando que no se repita su nombre en otros usuarios.
- **systemLogin:** Hace iniciar al usuario de sesión en el sistema, pero primero se comprueba que no esté "conectado", que no haya nadie con su misma id (su nombre) o que no existe una sesión ya iniciada por otro usuario.
- **systemLogout:** Permite cerrar la sesión anteriormente abierta por un mismo usuario.

- **systemTalk:** Función que permite al usuario interactuar con el chatbot, revisando previamente que el usuario haya iniciado sesión para poder interactuar.
- **systemSynthesis:** Entrega una síntesis del chatbot para un usuario en particular, esto es a partir del historial del chat del usuario que está en el respectivo sistema.
- **systemSimulate:** Permite que exista una simulación de interacción entre dos chatbots. De forma pseudoaleatoria, se da la posibilidad de que interactúen de forma “realista”, si no hay más interacciones, la simulación termina, esto es al igual con la cota superior, que es el número máximo de interacciones.

5. Diseño de la solución

Para empezar a diseñar la solución a la problemática se empezó planteando los TDA’s fundamentales para lograr el objetivo dado, para luego así pensar en las clases y sus métodos y atributos, para luego unirlos y crear el diagrama de análisis, pero luego, detallándolo más a profundidad, se obtiene el diagrama de diseño *[Figura 1]*. De esta forma, la primera clase por así decirlo fue el option, la base de todo, ya que este representa las opciones/respuestas que puede optar el usuario ante el chatbot, se constituye por un id único, un mensaje predeterminado que entregar, el link del id del chatbot asociado y del flow inicial y también posee una lista de las palabras claves, que serán necesarias para después. Esto fue necesario crearlo primero para poder crear el TDA flow ya que este se representa como un flujo en particular sin ningún tipo de repetición de un chatbot, este se representa por un id único de tipo entero positivo, un nombre del mensaje propiamente tal del flujo de tipo string y un listado de las posibles opciones a escoger, donde cada uno de los elementos es de tipo option. Siendo necesario verificar cada elemento ingresado para saber si es de acuerdo a lo solicitado y a los parámetros dados, de esta forma se puede lograr añadir opciones nuevas a un flujo, siempre y cuando las Id de las opciones no se repitan, que se puede verificar rápidamente creando un método que lo valide, ya que es necesario revisar listas en listas, si se repite un Id, no lo aceptará y no cambiará nada, este mismo procedimiento y pensamiento orientado a objetos se puede emplear en cada método similar donde se quiera añadir y verificar por medio de una Id. A continuación, se crea el TDA del chatbot como tal, este posee un id único, un nombre, un mensaje de presentación, el id del flow inicial asociado y una lista de los flujos, recordando de que cada elemento en cada TDA está representado por un *integer* positivo, un *string* o de tipo lista de otros TDA’s. De esta misma forma se crea el TDA system donde se alberga todo lo anterior, que por consiguiente es necesario crear el TDA usuario que representa a un usuario y el TDA de un historial del chat que se representa por mensaje.

Al tener todo ya establecido y representado se puede continuar a la creación del sistema de *login* y *logout* del usuario, donde a grandes rasgos es agregar al usuario (verificando duplicidad) a la lista de usuarios conectados en el sistema o quitarlo, de esta forma se puede crear el predicado *systemTalk*, donde se utilice las palabras claves de cada opción o utilizar sus Id respectivos para escoger una opción, siempre y cuando el usuario esté conectado (verificando por medio de su nombre como Id). Con esto terminado se puede obtener una síntesis de tipo string reseteable para cada conversación de cada usuario. De esta forma ya se podría tener una conversación completa con un sistema completo insertado por medio del main en Java, pero luego falta crear un menú interactivo por consola, donde se tiene que abstraerse la mirada al proyecto de tal forma para poder crear un menú que se conecte con cada respuesta esperada por parte del usuario.

6. Aspectos de la implementación

Para la implementación en este laboratorio se estructuró principalmente de 5 TDA's, siendo estos option, flow, chatbot, system, user. Estos TDA's fueron necesarios para la implementación total del ambiente esperado, ya que estos son los requerimientos principales obligatorios.

Cada uno de estos TDA's poseen sus respectivas clases, constructores, selectores, modificadores y otras operaciones para un buen y completo uso de estos mismos.

Se utilizó IntelliJ IDEA 2023.2.5 de 64 bits, versión 11.0.21 del jdk en lenguaje de Java, las bibliotecas empleadas fueron solamente las que vienen con IntelliJ IDEA y la biblioteca java.util.ArrayList.

7. Instrucciones de uso

Para poder utilizar y ejecutarlo por comandos, se requiere que se utilice por medio de los comandos de consola*, se utiliza Windows, para compilar: gradlew.bat build y para ejecución: gradlew.bat run, esto se puede escribir en el mismo cmd en la carpeta.*

8. Resultados

Probando con todos y cada uno de los métodos realizados (hasta *systemLogout*) y gran parte del main, cumplen su función en un 100%, pero lastimosamente no se pudo lograr finalizar por completo todo lo esperado, en especial a lo que se refiere con el main. Pero cada uno de los métodos hechos fueron probados con cada uno de las posibles complicaciones que podrían tener, como el *chatbotAddFlow* por ejemplo, o el *systemLogin*, que pueden mostrar errores fácilmente.

De los predicados no implementados fueron el systemTalk, systemSynthesis y el systemSimulate, que por falta de tiempo y de entendimiento sobre el paradigma para abordarlos, no se logró implementar.

9. Conclusiones

En conclusión me apena decir que no se logró lo esperado, la implementación en su totalidad del sistema de chatbots, pero cabe destacar el uso del paradigma presente, el paradigma orientado a objetos, ya que este tiene un gran alcance en demasiadas cosas, aunque no se necesite un nivel más de abstracción en comparación al paradigma lógico, de igual forma fue complicado.

En conclusión, se logró implementar eficientemente este paradigma en cada punto realizado, pero no fue terminado el proyecto en su totalidad, por la falta de familiaridad con el paradigma y el lenguaje Java. Lo más importante ante este paradigma es familiarizarse con la familia imperativa primeramente, y luego abordar el pensamiento orientado a objetos para así lograr unos buenos diagramas, de otra forma sería mucho más difícil el proceso de la elaboración del proyecto.

10. Referencias

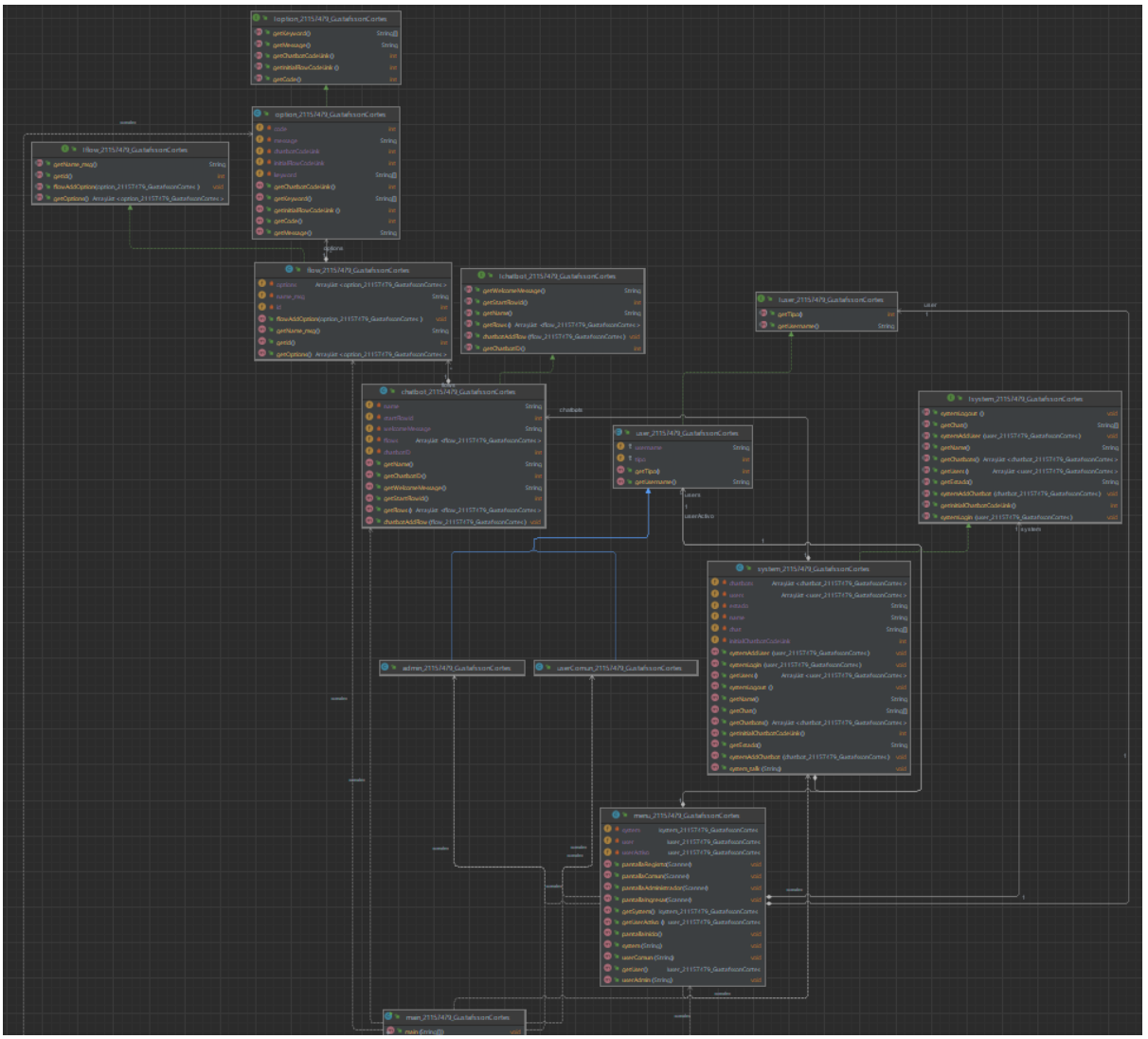
Gonzalez, Roberto. (2023). - Programación POO. Diapositivas/videos extraídos desde Moodle Usach - Paradigmas de Programación (13204 y 13310) 2-2023.

“Centro de ayuda de Java.” *Java*, https://www.java.com/es/download/help/index.html. Accessed 11 December 2023.

“Funcionalidades - IntelliJ IDEA.” *JetBrains*, https://www.jetbrains.com/es-es/idea/features/. Accessed 11 December 2023.

11. Anexos

[Figura 1]


Ioption_2115749_GustafssonCortes
option_2115749_GustafssonCortes
Iflow_2115749_GustafssonCortes
flow_2115749_GustafssonCortes
Ichatbot_2115749_GustafssonCortes
Iuser_2115749_GustafssonCortes
chatbot_2115749_GustafssonCortes
user_2115749_GustafssonCortes
Isystem_2115749_GustafssonCortes
system_2115749_GustafssonCortes
admin_2115749_GustafssonCortes
userComun_2115749_GustafssonCortes
menu_2115749_GustafssonCortes
main_2115749_GustafssonCortes